Available *online* at : http://bit.ly/InfoTekJar

InfoTekJar : Jurnal Nasional Informatika dan Teknologi Jaringan

ISSN (Print) 2540-7597 | ISSN (Online) 2540-7600

Computer Network, Server, Load Balancing

Analisis Perbandingan Algoritma *Static Round-Robin* dengan *Least-Connection* Terhadap Efisiensi *Load Balancing* pada *Load Balancer* Haproxy

Hasta Triangga [1], Ilham Faisal [2], Imran Lubis[3]

Teknik Informatika, Fakultas Teknik dan Komputer, Universitas Harapan Medan Jl. HM. Joni No.70 C Kota Medan, 20216 Sumatera Utara, Indonesia

KEYWORDS

Load Balancing, HAProxy, Static Round-Robin, Least connection, Reverse Proxy

CORRESPONDENCE

Phone: +62 813 62139818

E-mail: angga777@gmail.com

A B S T R A C T

In IT networking, load balancing used to share the traffic between backend servers. The idea is to make effective and efficient load sharing. Load balancing uses scheduling algorithms in the process includes Static round-robin and Least-connection algorithm. Haproxy is a load balancer that can be used to perform the load balancing technique and run by Linux operating systems. In this research, Haproxy uses 4 Nginx web server as backend servers. Haproxy act as a reverse proxy which accessed by the client while the backend servers handle HTTP requests. The experiment involves 20 Client PCs that are used to perform HTTP requests simultaneously, using the Static round-robin algorithm and Least-connection on the haproxy load balancer alternately. When using Static round-robin algorithm, the results obtained average percentages of CPU usage successively for 1 minute; 5 minutes; and 15 minutes are; 0.1%; 0.25%; and 1.15% with average throughput produced is 14.74 kbps. Average total delay produced 64.3 kbps. The average total delay and jitter is 181.3 ms and 11.1 ms, respectively. As for the Least-connection algorithm average percentage obtained successively for 1 minute; 5 minutes; and 15 minutes are 0.1%; 0.3%; and 1.25% with the average throughput produced is 14.66 kbps. The average total delay and jitter is 350.3 ms and 24.5 ms, respectively. It means Static round-robin algorithm is more efficient than the algorithms Least-connection because it can produce a greater throughput with less CPU load and less total delay.

PENDAHULUAN

HAProxy merupakan software *load balancer* yang cukup banyak digunakan karena memiliki lisensi opensource dan berjalan pada sistem operasi linux. HAProxy yang dikonfigurasi sebagai *load balancer* dengan menggunakan dua atau lebih server backend dapat menambah kemampuan sistem untuk menangani request dari client [1].

Untuk melakukan load balancing, load balancer menggunakan beberapa algoritma penjadwalan antara lain round-robin dan least-connection. Dengan algoritma tersebut *laod balancer* dapat meneruskan paket permintaan dari pengguna ke server. Misalnya, dengan mengunakan algoritma *static Round-robin, load balancer* dapat mengirimakan paket koneksi baru yang masuk secara bergantian dan stimultan ke masing-masing server backend. Serta ada juga algoritma *least-connection* yang dapat mentukan arah lalu-lintas koneksi yang datang ke dalam *cluster* berdasarkan jumlah koneksi aktif masing-masing server *backend* [2].

Untuk Mengukur tingkat efisiensi dari load balancing dibutuhkan perhitungan quality of service (QOS) khususnya *throughput, delay; Jitter* yang dapat memeberikan hasil yang akurat, karena QOS dapat menetukan karakteristik dari koneksi dalam sebuah jaringan [3]. Oleh karena itu penelitian tentang Analisis Perbandingan Algoritma Static Round-robin dengan Least-connection Terhadap Efisiensi Load Balancing Pada Load Balancer HAProxy menarik untuk dilakukan.

Variabel yang akan dihitung pada penelitian kali ini antara lain: CPU; *Throughput*; *Delay*; dan *Jitter*. Untuk nilai CPU, dan *throughput* didapat menggunakan *server monitoring* Zabbix. Sedangkan untuk nilai *delay* dan *jitter* didapat menggunakan aplikasi *network packet analyzer* Wireshark yang selanjutnya dilakukan penghitungan menggunakan rumus *delay* dan *jitter*.

Adapun tujuan penelitian ini yaitu menentukan algoritma *static round-robin* atau *least-connection* yang akan diaplikasikan pada *load balancer* HAProxy sehingga diperoleh pembagian beban lingkungan *cluster* server *backend* yang lebih efisien.

https://doi.org/10.30743/infotekjar.v4i1.1688

HAProxy

HAProxy merupakan produk software yang dikembangkan untuk melakukan proses load balancing dan failover, dengan kata lain, apabila terjadi kegagalan pada satu node dalam jaringan cluster, Haproxy akan mengalihakan ke node lainnya yang tersedia di cluster tersebut. HAProxy berkerja pada sistem operasi Linux dan memiliki lisensi open source sehingga aplikasi ini dapat dengan bebas dikembangkan oleh para pengembang aplikasi [1].

Untuk melakukan proses *load balancing*, sebuah *load balancer* membutuhkan suatu metode dalam proses perhitungannya, disebut *scheduling*. Ada 8 (delapan) algoritma *scheduling* HA*Proxy* yang dapat digunakan dalam konfigurasi, yaitu: *Round-Robin; Static Round-Robin; Least-Connection; Source; URI (Uniform Resource Identifier); URL_parameter; Header Name; dan RDP Cookie*, Namun, fokus dalam penelitian kali ini pada algoritma *Static Round-Robin dengan Least-Connection.* Dalam melakukan konfigurasi HA*Proxy*, ada bagian-bagian yang perlu diperhatikan, yaitu: *Global settings; Default Settings; Fronted Settings; dan Backend Setting.* Untuk Algoritma *scheduling* yang digunakan pada HA*Proxy* terdapat pada konfigurasi *Backend setting* [4], sedangkan algoritma yang akan digunakan pada penelitian kali ini adalah adalah algoritma *static round-robin* dan *least connection.*

Algoritma Static Round-Robin

Algoritma *Static Round-robin* atau merupakan algoritma yang bekerja dengan cara mendistribusikan tiap *request* yang datang ke *pool* sever-*server backend* yang sebenarnya, mirip dengan algoritma *round-robin* DNS. Pada algoritma ini semua *node server* akan diperlakukan setara sesuai dengan beban yang telah ditetapkan masing-masing *server*, namun tidak mengizinkan peralihan beban secara dinamis dikarenakan sifat alami beban statis. Tidak ada batasan jumlah *server* aktif pada *backend*.

Algoritma Least-Connection

Algoritma *Least-Connection* bekerja dengan cara mendistribusikan lebih banyak permintaan (*request*) ke *server* asli berdasarkan koneksi aktif yang lebih sedikit, selain itu algoritma ini mengganggap semua *server backend* memiliki kemampuan komputasi yang sama [2]. Algorima *Least Connection* cocok untuk digunakan pada *cluster* dengan lingkungan yang dinamis dengan *session* yang berubah-ubah [4]. Untuk menggunakan Algorima ini, perintah *balance leastconn* harus disertakan pada konfigurasi *backend settings.*

Bandwidth dan Throughput

Throughput merupakan jumlah bit yang diterima dengan sempurna perdetik pada sebuah perangkat dalam sebuah proses pengiriman data. Sedangkan *bandwidth* merupakan batasan banyak nya paket data yang bisa dilewati oleh suatu perangkat per satuan detik [5] [6].

Delay dan Jitter

Delay (*latency*) merupakan waktu yang dibutuhkan paket data untuk tiba tujuan dalam sebuah proses pengiriman data. *Latency* dipengaruhi oleh beberapa faktor antara lain: jarak; waktu proses; dan media [3]. *Delay* dapat dihitung menggunakan persamaan berikut[3][7].

$$delay = \frac{PacketLength}{Linkbanwidth}$$

$$delay = t_n - t_{n-1}$$

Jitter merupakan variasi delay rata-rata yang diakibatkan banyak kejadian dalam suatu proses pengiriman data seperti panjang antrian dan penghimpunan ulang paket-paket data [3]. Jitter dapat dihitung dengan persamaan berikut.

$$Jitter = \frac{TotalVariasiDelay}{TotalPaketData - 1}$$

$$TotalVariasiDelay = \sum | delay_n - delay_{n-1} |$$

Dengan menghitung Jitter dan delay maka kualitas suatu layanan dalam jaringan dapat diukur [3]. Pembagian kategori Kualitas suatu jaringan dapat dilihat pada tabel berikut.

Tabel 1 Indeks Kualitas Layanan Jitter[3]

Indeks	Nilai Jitter (ms)	Kategori
4	0 ms	Sangat baik
3	0 ms s/d 75 ms	Baik
2	75 ms s/d 125 ms	Sedang
1	125 ms s/d 225 ms	Buruk

Tabel 2 Indeks Kualitas Layanan Delay[3]

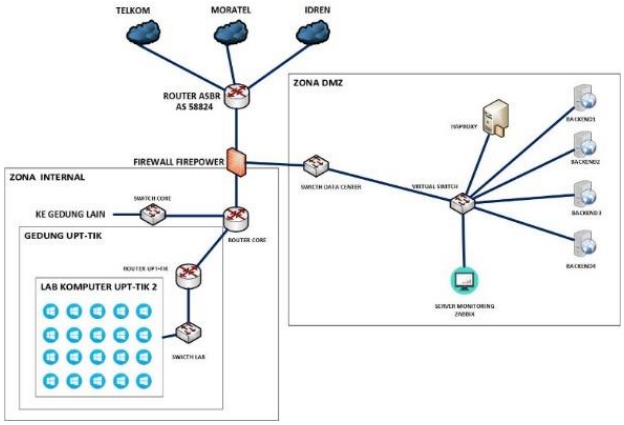
Indeks	Besar delay (ms)	Kategori
4	< 150 ms	Sangat baik
3	150 ms s/d 300 ms	Baik
2	300 ms s/d 450 ms	Sedang
1	>450	Buruk

METODOLOGI

Untuk mencapai tujuan penelitian, maka penelitian direncanakan secara sistematis untuk mendapatkan nilai yang seobjektif mungkin, antara lain dengan melakukan perancangan topologi, kemudian masing-masing variabel pada masing-masing algoritma dihitung dan dibandingkan untuk melihat algoritma yang paling efisien antara dua algoritma tersebut. Adapun variabel yang akan diuji adalah: *(1) CPU Jumps (2) Throughput (3) delay (4) Jitter*. Sedangkan Bandwidth yang digunakan pada masing-masing adalah 1Gbps karena menggunakna koneksi *intranet*. Adapun langkah-langkah yang akan dilakukan pada penelitian ini adalah sebagai berikut.

Menentukan Topologi

Penelitian ini melibatkan 20 PC *client* untuk melakukan *request* ke *server*, sedangkan *load balancer* HAProxy dan server *backend* berada di area DMZ (*demiliterized zone*). Rancangan topologi digambarkan dengan tujuan untuk mendapatkan informasi visual tentang tata letak dari perangkat yang digunakan.



Gambar 1 Topologi Penelitian

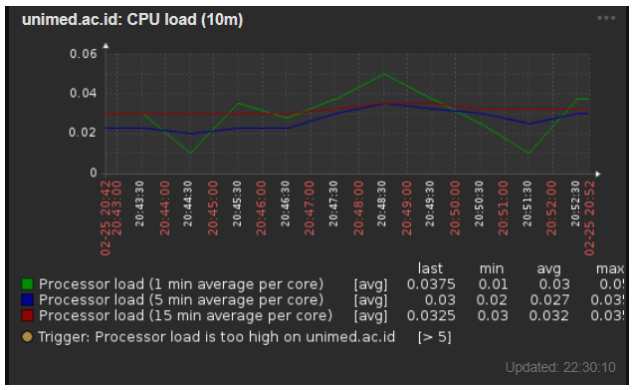
Setelah topologi ditentukan, maka selanjutnya adalah memberikan alamat IP pada masing-masing server. Untuk alamat IP komputer client menggunakan DHCP yang sudah ada pada gedung UPT-TIK Universitas Negeri Medan. Berikut adalah datar IP masing-masing perangkat yang digunakan.

Tabel 3 Konfigurasi IP masing-masing server

Mesin	IP address	Subnetmask	Default Gateway
client	15.50.1.2.254	255.255.255.0	15.50.1.1
haproxy	192.168.250.222	255.255.255.0	192.168.250.254
backend 1	192.168.250.231	255.255.255.0	192.168.250.254
backend 2	192.168.250.232	255.255.255.0	192.168.250.254
backend 3	192.168.250.233	255.255.255.0	192.168.250.254
backend 4	192.168.250.234	255.255.255.0	192.168.250.254
Server monitor	192.168.250.138	255.255.255.0	192.168.250.254

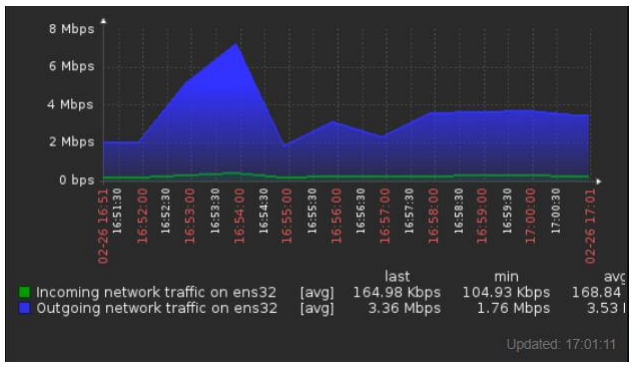
Proses Pengujian

Pengujian kedua algoritma menggunakan 20 PC client dengan cara melakukan *request* HTTP secara serentak dari 20 PC ke *server load balancer* Ha*proxy*. data berasal dari *server monitoring* Zabbix. Data yang diambil nantinya adalah data kejadian terakhir pada grafik CPU dan *Network traffic*. Sedangkan banyaknya pengujian yang dilakukan adalah sebanyak 5 (lima) kali, hal ini dilakukan agar data yang diperoleh lebih akurat.



Gambar 2 Grafik CPU Jump Server Haproxy Menggunakan Zabbix

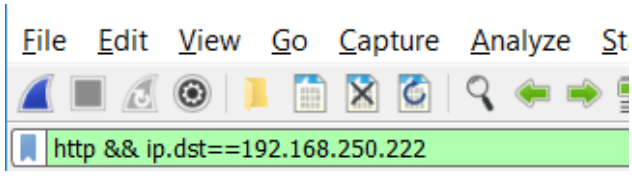
Gambar 2 menunjukkan grafik pengguanaan *CPU* pada *server*, beban CPU rata-rata pada 5 menit 10 menit dan 15 menit. Data yang diolah nantinya adalah data *last*. Data *last* adalah waktu perkiraan saat server menerima *request* HTTP dari *client*. Data yang akan diperoleh dalam bentuk desimal dan akan diubah dalam bentuk persen.



Gambar 3 Grafik *throughput interface* server

Gambar 3 menunjukkan grafik *throughput interface* jaringan. Nilai yang akan digunakan nantinya adalah nilai *incoming network traffic* karena merupakan gafik arus data (*throughput*) yang diterima *server* dari *user*. Sedangkan *outgoing network traffic* adalah arus data yang berasal dari *server* ke *user*.

Variabel selanjutnya yang akan diuji adalah *delay* dan *jitter*, untuk mendapatkan nilai variabel tersebut, terlebih dahulu dilakukan monitoring paket data yang menuju haproxy menggunakan wireshark. Untuk mendapatkan paket HTTP ke server haproxy, filter paket dilakukan dengan perintah berikut sehingga diperoleh variabel waktu masing-masing *request*.



Gambar 4 String Filter Koneksi HTTP Ke Server Haproxy

 https://doi.org/10.30743/infotekjar.v4i1.1688

No.	Time
48	3.413414
52	3.428090
53	3.429704
57	3.449195
59	3.455097
60	3.455664
63	3.470337
64	3.470432
66	3.514525
83	3.565498
87	3.595993
88	3.613551

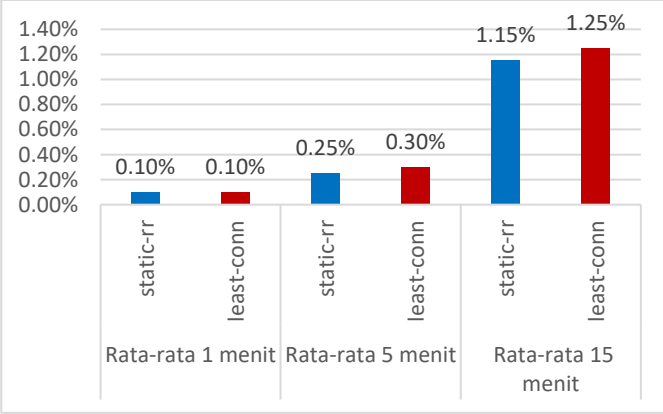
Gambar 5 Variabel *Time* Dari Pengujian Menggunakan Wireshark

Setelah variabel waktu didapat, maka langkah selanjutnya adalah menghitung *delay* menggunakan persamaan 2 dan *jitter* menggunakan persamaan 3 dan 4. Sehingga didapat nilai kualitas layanan (*quality of service*) pada masing-masing algoritma.

Jika semua variabel didapat, maka hasil pengujian dengan algoritma *static round-robin* dibandingkan dengan hasil pengujian menggunakan algoritma least-connection. Algoritma yang mendapatkan nilai paling rendah atau kategori QOS yang lebih baik adalah algoritma yang paling efisien untuk digunakan pada *load balancer* HAProxy.

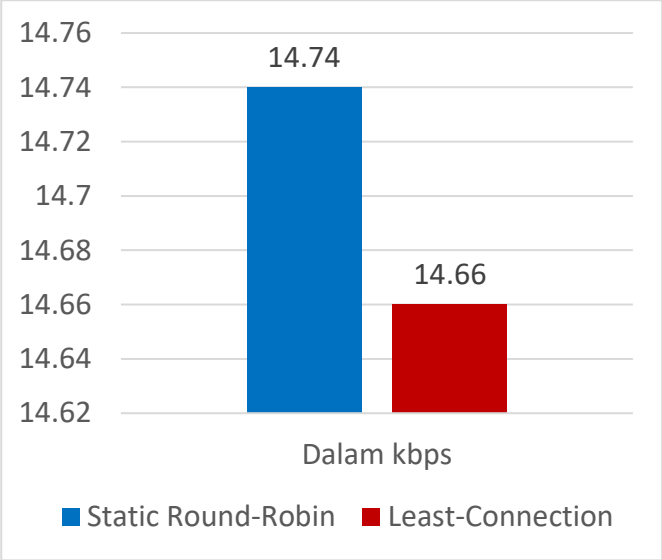
Hasil dan Pembahasan

Hasil yang diperoleh dari pengujian *CPU, Throughput, delay dan Jitter* dengan menggunakan algoritma *static round-robin* dan *least-connection* adalah sebagai berikut.



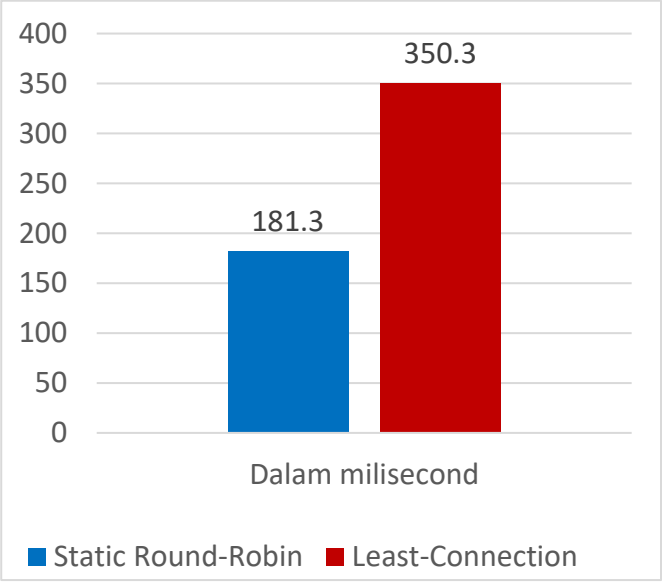
Gambar 6 Grafik Perbandingan *CPU Load* Antara Algoritma Static-Rr Dengan Algoritma *Least-Conn*

Dari gambar 6 terlihat bahwa penggunaan beban CPU *server* ha*proxy* dengan menggunakan algoritma *static round-robin* dan *least-connection* sangat minim, baik rata-rata 5 menit, 10 menit, dan 15 menit. Beban CPU bahkan tidak mencapai angka 2 % per *core* –nya dengan pengujian 20 PC. Namun terdapat sedikit perbedaan pada masing-masing waktu rata-rata beban CPU. Terlihat ketika *load balancer* menggunakam algoritma *static round-robin*, penggunaan CPU lebih sedikit 0.05% dibandingkan dengan ketika server menggunakan algoritma *least-connection*. Perbedaan lebih terlihat pada waktu rata-rata 15 menit beban CPU, algoritma static round-robin lebih rendah 0.1% jika dibandingkan dengan pengunaan algoritma *least-connection*. Sedang pada *throughput* didapat hasil sebagai berikut.



Gambar 7 Grafik Perbandingan Throughput Algoritma *Static Round-Robin* Dengan Least-Connection

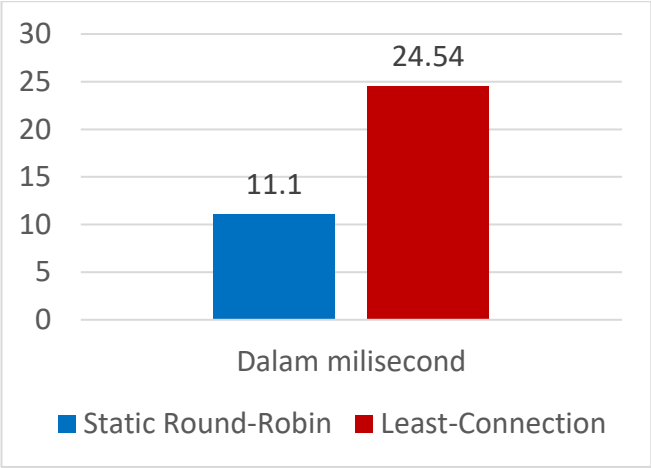
Pada gambar 7 juga terlihat bahwa penggunaan algoritma *static-rr* lebih tinggi *throughput*-nya 14.74 bernading 14.66 kbps walaupun penggunaan CPU nya lebih kecil.



Gambar 8 Grafik Perbandingan Total *Delay*

Pada gambar 8 terlihat nilai Total delay pada algoritma *Static Round-Robin* lebih kecil. Dengan nilai 181.3 ms, pengujian *delay* pada algoritma static-rr masih dalam kategori baik. Sedangkan untuk *Least-connection* sudah masuk kategori sedang karena total *delay* berada pada kisaran 300 – 450 ms. Pada variabel ini penggunaan algoritma Static *Round-Robin* lebih efisien dibandingkan dengan penggunaan algortima *Least-Connection*.

Hasil Pengujian berikutnya yang didapat adalah *Jitter*. Ternyata, Jitter yang diperoleh menggunakan algoritma *static round-robin* sedikit lebih rendah dibandingkan dengan *least-connection*.



Gambar 9 Grafik Perbandingan Jitter

Pada variabel Jitter, nilai yang diperoleh dari penelitian untuk kedua algoritma masih dikisaran baik karena masih berada dikisaran 0-75 ms. Artinya penggunaan load balancing dengan algoritma Static Round-Robin lebih efisien dibandingkan dengan penggunaan algoritma *Least-Connection* pada *load balancer* HAProxy. Jika dibandingkan secara keseluruhan variabel. Maka hasilnya akan terlihat seperti pada tabel berikut.

Tabel 4 Perbandingan Hasil Pengujian

	Static-rr	**Kotegori QoS**	**Least-conn**	**Kategori Qos**
CPU Rata-rata 1 menit	0.10%	-	0.10%	-
CPU Rata-rata 5 menit	0.25%	-	0.30%	-
CPU Rata-rata 15 menit	1.15%	-	1.25%	-
Throughput	14.74 ms	-	14.66 ms	-
Total Delay	181.3 ms	baik	350.3 ms	sedang
Jiter	11.1 ms	baik	24.54 ms	baik

Pada tabel 4 dapat terlihat persentase CPU yang digunakan sangat kecil, namun sedikit terlihat bahwa penggunaan algoritma least-connection sedikit lebih tinggi. Sedangkan nilai throughput yang didapat untuk algoritma static round-robin sedikit lebih besar dibandingkan dengan penggunaan least-connection yaitu 14.74 kbps; 14.66 kbps. Namun tidak begitu mencolok karena pengujian yang dilakukan hanya menggunakan 20 (dua puluh) request dari 20 PC client. Untuk variabel delay dan jitter yang diuji, terlihat bahwa total delay dan jitter yang didapat dari penelitian dengan menggunakan algoritma Static Round-Robin lebih kecil yaitu berturut-turut adalah 181.3 ms berbanding 350 ms, dan 11.1 berbanding 25.54 ms untuk nilai *jitter* nya.

KESIMPULAN DAN SARAN

Berdasarkan penelitian yang telah ditentukan dapat disimpulkan bahwa penggunaan algoritma *static round-robin* lebih efisien. Hal ini ditunjukkan oleh grafik *CPU load* dimana ketika menggunakan algoritma *static round-robin* nilai rata-rata yang diperoleh untuk beban CPU sedikit lebih kecil dan *throghput* lebih besar dibandingkan dengan penggunaan algrotima *least-connection*, nilai yang diperoleh untuk beban CPU adalah 0.1 %; 0,25%; dan 1.15%; untuk rata-rata 1 menit; 5 menit; dan 15 menit; dan nilai *throughput* rata-rata adalah 14.74 kbps. Nilai total *delay* dan *jitter* rata-rata yang diperoleh dari percobaan dengan menggunakan algortima *static round-robin* juga lebih kecil yaitu masing-masing 181.3 ms dan 11.1 ms. Nilai hasil *Jitter* dan Total *Delay* pada algoritma *Static Round-Robin* masih masuk dalam kategori baik pada tabel quality of services.

Perbandingan nilai beban CPU dan *throughput* tidak terlalu terlihat pada hasil penelitian kali ini karena hanya menggunakan 20 PC *client* untuk melakukan *request* http. Pada penelitian berikut nya yang berkaitan dengan hal ini, sebaiknya.

a. Pengujian dilakukan dengan cara *stress test* dengan volume yang lebih besar atau *network benchmarking* agar perbedaan penggunaan algoritma *static round-robin* dan *least-connection* pada ha*proxy* lebih terlihat.
b. Jika menggunakan *load balancer*, maka integritas data dari tiap *server backend* harus tetap terjaga dengan baik sehingga dibutuhkan penelitian lebih lanjut tentang integritas data pada masing-masing *server backend*.

UCAPAN TERIMA KASIH

Terima kasih kepada Allah Subhanahu wa ta'ala yang telah meridhai kami untuk menulis penelitian ini, dan juga nabi Muhammad Sallallahu 'Alaihi Wasallam yang telah membawa ajaran agama islam dan menjadi contoh teladan terbaik. Terima kasih juga untuk pada dosen pembimbing jurnal yang telah membimbing saya dalam penulisan karya ilmiah ini. Kepada anak dan istri saya sebagai motivasi tambahan. Dan juga teman-teman di UPT-TIK terutama abangda Mohamad Ihwani dan Oris Krianto Sulaiman yang ikut membantu dalam penelitian ini.

1. REFERENSI

[1] A. B. Noviyanto, E. Kumalasari, and A. Hamzah, "Perancangan dan Implementasi Load Balancing Reverse Proxy Menggunakan HAProxy pada Aplikasi Web," *J. Jar. Komput.*, vol. 3, no. 1, pp. 59–68, 2015.

[2] M. E. Mustafa, "Load Balancing Algorithms Round-Robin (Rr), Least- Connection , and Least Loaded Efficiency," vol. 1, no. 1, pp. 25–29, 2017.

[3] R. Wulandari, "Analisis Qos (Quality Of Service) Pada Jaringan Internet (Studi Kasus : Upt Loka Uji Teknik Penambangan Jampang Kulon – LIPI)," *J. Tek. Inform. dan Sist. Inf.*, vol. 2, no. 2, pp. 162–172, 2018.

[4] S. Levine and S. Wadeley, *Red Hat Enterprise Linux 7*. Red Hat, Inc. and others., 2016.

[5] H. Nasser and T. Witono, "Analisis Algoritma Round Robin, Least Connection, Dan Ratio Pada Load Balancing Menggunakan Opnet Modeler," *J. Inform.*, vol. 12, no. 1, pp. 25–32, 2018.

 https://doi.org/10.30743/infotekjar.v4i1.1688

[6] G. F. Ardiansa Eko, R. Primananda, and M. H. Hanafi, "Manajemen Bandwidth dan Manajemen Pengguna pada Jaringan Wireless Mesh Network dengan Mikrotik," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 11, pp. 1226–1235, 2017.

[7] J. D. Danur, "Analisa Kinerja Jaringan Provider untuk Aplikasi Video Chatting (Studi Kasus di daerah Marpoyan)," *Jom FTEKNIK Jur. Tek. Elektro Univ. Riau*, vol. 3, no. 2, pp. 1–8, 2016.

2. BIOGRAFI PENULIS

Hasta Triangga
Lulusan Universitas Harapan Medan pada april 2019. Bekerja di Universitas Negeri Medan sebagai System Administrator sejak 2013 sampai dengan karya ilmiah ini dibuat.